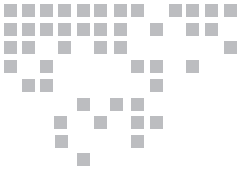
第一部分 Part 1

起　步



Chapter 1 第 1 章

术语与最佳实践

虚拟现实（VR）开发的世界之所以这样令人生畏，是因为当前有大量可用的竞争硬件与软件。当然，也因为 VR 是新媒介，很多游戏开发者公认的原理与体验在 VR 游戏中并不可用。

如果你不能区分 Oculus VR、OSVR 和 OpenVR，或者正在为开启 VR 之旅寻找一些通用最佳实践，那么本章内容将对你很有帮助。

1.1 术语

许多技术、软件和设备构成了日益增长的 VR 生态系统。为了保证在同一水平上，我们先来看看这生态系统的几个关键部分，这些部分是任何 Unreal Engine 4（UE4）VR 开发者都应该知道的内容。如果已经熟悉 VR 产业的现状以及相关技术的读者，请自动跳过本章。

1.1.1 设备

在创建体验的时候，可以在大量的硬件设备中选择，无论其是一个 VR Head

Mounted Display（HMD）还是一个 VR-ready 控制器。Unreal Engine 支持这些设备中的绝大多数，这减轻了开发人员在项目初始阶段选择正确设备的负担。UE4 当然有很好的 VR 抽象层，所以决定改变目标设备（或者在管道的任何阶段适配多目标设备）是很容易的。

原生支持的 HMD 如表 1-1 所述。

表 1-1　支持的 VR HMD

HMD	描　　述
Samsung Gear VR	Gear VR 是兼容 Samsung 智能手机的硬件产品，是 Oculus 为了在移动 VR 软件方向有更好的发展而与 Samsung 合作开发的。当使用 Gear VR 作为目标硬件时，可以使用 Oculus Mobile 软件开发包（SDK）来获得这些软件的特性。Gear VR 目前只支持虚拟头部模型偏移的旋转追踪
HTC Vive	HTC Vive 的开发与 Gear VR 的合作模式类似，只不过是 HTC 提供硬件，Valve 提供软件 SDK，它们都是基于 SteamVR 和 OpenVR。Vive 提供旋转追踪，也就是为头盔和运动控制器提供一对一位置追踪。使用 Vive，Valve 提供 OpenVR SDK（更多关于 OpenVR的信息，参阅表 1-3），以便访问 VR 体验所需的各种传感器和设置数据
Oculus Rift	Oculus Rift 以开发者版本的形式经历了许多不属于消费者的迭代。在 2016 年，Oculus 发布了 Consumer Version 1（CV1），它面向消费者并可以购买设备。CV1 是一个完全集成了 Oculus 硬件与软件 SDK（表 1-3 可以看到 Oculus SDK）的系统。Rift 提供与 HTC Vive 类似的旋转与位置追踪。它的默认输入设备是 Xbox One 控制器和 Oculus Remote；Oculus Touch 运动控制器（见表 1-2）是和本体分开发货的
Google Cardboard	Google Cardboard 很小，通常 5 ～ 20 美元就可以买到，它包含一个简单的按钮和镜头。它允许你把手机放在上面，并展示立体内容。Cardboard 依靠智能手机内置的传感器来检测用户的头部旋转（如果不校准，其可能不准确）。它也依赖于原生 Android SDK 提供的大多数设备的兼容性（这可能比其他解决方案有更高的渲染延迟）。但是，Cardboard 是迄今为止最便宜的 HMD
Google VR/Daydream VR	Google VR 或者 Daydream VR 是比其他方案（Google Cardboard）更为严格的控制软件与硬件生态系统。对于硬件，无论设备（手机）还是头盔都必须以“Daydream Ready”归类，这样 Google 可追加更多高级的软件特性。Daydream 同样支持 Daydream 控制器（见表 1-2），它提供触摸板和旋转追踪。用 Daydream 设备，可以使用 Google VR SDK（见表 1-3）
PlayStation VR	PlayStation VR 是在 PlayStation 4 和 PlayStation 4 Pro 游戏机上玩 VR 游戏的外围设备。位置追踪是通过扩展 PlayStation Camera 获得的，旋转追踪是通过使用 IMU（惯性测量单元）获得，功能类似于表中其他的头盔。玩家也可以使用 PlayStation Move 控制器（见表 1-2）来获得运动控制器的全部体验。DualShock 4 也可以使用；它也具有位置跟踪的能力
OSVR	Open Source Virtual Reality(OSVR）是来自几个行业的倡议。目前，OSVR 在 HDK1.3 和 HDK2 上提供 Hacker Development Kit（HDK），通过 Razer 开发。两种头显都提供旋转和位置追踪功能

在 UE4 中，运动控制器是通过一个运动控制器组件来支持的，它可以很容易地适配多控制器。UE4 支持的运动控制器的相关细节见表 1-2。

表 1-2 支持的运动控制器

运动控制器	描 述
Oculus Touch	Touch 控制器通过 Oculus 的 Constellation 追踪系统，可以提供旋转和位置追踪。每个控制器都有两个按钮（A、B、X 和 Y，组成每个控制器的十字形）、一个拇指摇杆、一个食指扳机以及一个手部扳机。通过 Oculus SDK 可以支持 Touch 控制器
Vive	Vive 控制器通过 Valve 的 Lighthouse 追踪系统，可以提供旋转和位置追踪。每个控制器包含一个圆形触控板，当其被按下时可以模拟输入。Vive 控制器提供扳机、菜单按钮和握紧按钮，通过 Open VR SDK 可以支持 Vive 控制器
PlayStation Move	Move 控制器通过 PtayStation Camera 和控制器顶部的 RGB 彩球，提供旋转与位置追踪。像其他 PlayStation 控制器一样，每个控制器都有十字形、圆形、三角形和方形按钮。它也有与扳机（或 T 按钮）类似的 Move、Start 和 Select 按钮
Daydream	Daydream 控制器只提供旋转追踪。它也提供了触摸板（其可以像按钮一样交互）和在它下面的个人 APP 按钮。通过 Google VR SDK 可以支持 Daydream 控制器

1.1.2 软件

有很多的 SDK、库和 API（应用程序编程接口）可以帮助连接 VR 硬件。UE4 倾向于把这些抽象成单个的接口或者组件，以实现简单的互操作性；但是，如果需要，也可以手动使用那些 SDK。对于开发人员来说，了解这些软件中不同的设计理念是很有用的，因为在开发游戏时可能需要使用 SDK 中的一些高级特性。手动使用这些 SDK，不需要下载任何文件；在你下载引擎时 UE4 包含了它们。

这些原生支持的 SDK 如表 1-3 所述。

表 1-3 支持的 SDK

SDK	描 述
Oculus PC SDK	Oculus PC SDK 提供 UE4 渲染适当视图所需的引擎信息。其包含一些内容比如用户头部位置、旋转量以及用户的 IPD（瞳孔间距离）。它还允许 UE4 提交渲染给 Oculus 合成，并纠正 HMD 的镜头畸变。幸运的是引擎已经做了这些事情，故而在大多数情况下不需要担心。Oculus SDK 还提供分层，这样可以在不同分辨率的屏幕上画不同的东西；这时用户接口（UI）很有用，你可能希望渲染一个比背景分辨率高的文本。ATW（Asynchronous Time-warp，见表 1-4）等功能在运行时是自动处理的，这里不用担心它是否已经开启了
Oculus Mobile SDK	Oculus Mobile SDK 可以访问很多与 PC SDK 一样的东西，比如头部的朝向或者其他需要渲染的用户信息。与 PC 运行时类似，移动运行时也使用 ATW；它也提供前缓冲区渲染，也叫线性扫描（见表 1-4）

（续）

SDK	描　　述
Oculus Audio SDK	Oculus Audio SDK 提供一种空间化的实现，为 VR 创建 3D 声音。它可以给声音资源应用 HRTF（head-related transfer function）以模拟定向音频（通过人耳模型建模）。它也可以通过衰减响度来模拟声音到耳朵的距离。UE4 中已经实现了 Oculus Audio SDK 的 HRTF，在引擎中可以手动创建距离曲线；但是，目前只实现基于 DirectX 的项目（比如 PC）
OpenVR SDK	OpenVR SDK 与 Oculus PC SDK 类似，也可以访问 HMD（比如位置和朝向）需要的信息。OpenVR SDK 也提供访问 Chaperone（Valve 指代用户可玩区域的虚拟边界）和类似 Oculus 分层的覆盖 2D 渲染内容的排序。OpenVR 也提供二次投影（类似于 ATW）和预测
OSVR SDK	OSVR SDK 可以访问创建 VR 体验的各种接口。因为天生“支持一切”的特性，它包含更多获得 HMD 信息的必要接口，比如位置和朝向，也包含一些关于眼睛追踪、万向跑步机和手势等的接口。OSVR SDK 已经整合到 Unreal Engine 4.12 中
Google VR SDK	Google VR SDK 可以访问 HMD 头部追踪数据，也有很多特性来优化体验。它可以开启持续运行模式，这对于手机很重要，因为对手机热度限制会导致很差的用户体验。其也可以访问线性扫描功能，可以优化渲染管线中的延迟（见表 1-4）

UE4 除了使用 SDK 和库与大量 VR 硬件交互外，在引擎内其还实现了一些特别的软件功能用于增加 VR 的体验。此外，其还实现了大量的运行时软件功能（比如 ATW），它们默认是开启的，方便开发者使用。在 UE4 中这两种类型的软件如表 1-4 所示。

有很多特性会在第 10 章中进一步展开。

表 1-4　支持的软件功能

特　　性	描　　述
实例化立体渲染	实例化立体渲染，是指同时渲染玩家视角左右眼，而不是逐个渲染，这会增加 GPU 和 CPU 的最大渲染时间。UE4 原生支持此特性，但默认是关闭的。因为某些渲染特性在这种模式下无法工作。由于某些特殊原因，实例化立体渲染也许会导致更糟糕的 GPU 表现，所以确定使用前一定要进行测试
隐藏与可见物体优化	隐藏区域的网格物体在 UE4 中允许渲染器对它们进行优化，从而避免 HMD 设备所导致的扭曲，来达到节约 GPU 渲染时间的目的。这些优化默认是开启的
时间扭曲 / 重投影	时间扭曲（使用 Oculus 运行时）和重投影（使用 OpenVR 运行时）允许渲染器弥补丢帧（当程序在下次刷新前没有足够的时间来渲染下一帧的内容时）而不是在同一帧展示相同的内容两次，从而导致画面颤抖。运行时将前一帧与最近的头部旋转进行重投影，来提供一种更为顺滑的体验（这两种绝妙的技术都会在第 10 章进行演示）。如上所述，这些功能都在运行时的不同时刻生效，所以不需要去主动开启它们。注意，开发者不要依赖这些特性来弥补坏帧率，而应该把它们作为自己无法控制之外的一些效能提升

（续）

特 性	描 述
前缓冲区渲染 / 线性追踪	当我们渲染常规 3D 应用的时候，为了避免将不完全的图片显示给用户，图片在渲染完成前放在后缓冲区，渲染完成后和前缓冲区进行交换（为了发送给屏幕）。这会对 VR 应用带来非常小的一帧的浪费（第 10 章有更详细的解释）。为了避免这种情况，主流的（目前是手机）开发工具（SDK）直接使用前缓冲区进行渲染（线性逐像素进行追踪，一行一行地从前缓冲区读取到屏幕上，称为“线性扫描”）。这种做法能成功的原因是一些 VR 帧渲染的可预测技术和重投影技术已经实现了。Oculus 移动设备开发包原生支持这些特性并默认开启，但在 Google VR 中你需要自己去开启这个选项
后向 / 前向渲染	在渲染 3D 场景时有两种非常重要的技术：前向与后向。刚开始大多数的游戏引擎都是基于前向流程，因为在渲染小场景时，它非常简单有效。但现在许多引擎都转向或者包含了后向渲染流程，因为它能允许更多动态光照和一些高级屏幕空间特效诸如 SSR(屏幕空间反射)。由于 VR 应用的画面表现极其重要，前向渲染是唯一选择，所以 Gear VR 和 Google VR 都是前向的。而到桌面端，UE4 默认是后向渲染的。然而，其也在积极地推进前向渲染流程的开发（目前已实现并推出。——译者注）

1.1.3 Unreal Engine

制作游戏和体验需要许多不同的系统共同工作；幸运的是，UE4 对这些系统有大量的工具。本书中使用的系统如表 1-5 所述。

表 1-5 本书使用的 Unreal Engine 系统

系 统	描 述
Unreal Motion Graphics (UMG)	在 UE4 中，UMG 是数据驱动的 UI 系统。为了使用 UMG Widgets，你可以使用 UMG UI Designer，其不需要任何一行代码就可以很容易地创建 UI。UMG 还支持动画关键帧和动态 UI 缩放
Blueprints	Blueprints（蓝图）是 UE4 的可视化脚本语言。它可以快速迭代，是一种避免陷入语言语法的好方法（本书中几乎只使用 Blueprints）
Sequencer	Sequencer 是 UE4 的高级电影工具。它可以通过关键帧创建一系列动画。可以在一个简单的类似传统 NLE（Nonlinear Editor，非线性编辑器）中剪辑和移动它们
Persona	Persona 是 UE4 的动画编辑器。它有很多功能，可以很轻松地混合动画用来创建反向运动学系统

1.2 最佳实践

VR 是一个新兴产业。在我们采取某种方法之前，必须进行大量的实验。很多时候，你最初只是对如何在 VR 中很好地运行某个机制或功能的想法很感兴趣，同时，

当最终实现并尝试它们时，你从未想过的事情能够很好地发挥作用。

话虽如此，但通过研究和试验，我们发现无论是使用 Oculus 还是 Epic，多数人在 VR 中都会出现不舒服的情况。在撰写本文时，参考文档地址如下：https://developer.oculus.com/documentation/intro-vr/latest/ ；https://docs.unrealengine.com/latest/INT/Platforms/VR/ContentSetup/index.html。

UE4 最佳实践对任何初始 Unreal Engine VR 开发者都很适用，因为很多的主题都具体到 UE4，以及如何优化 VR 项目。还有很多东西，比如禁用 HZB（Hierarchical Z-Buffer）和立体渲染，在第 10 章会详细介绍。如果想要了解更多 VR 体验的通用技巧，可以在 Oculus 最佳实践手册中找到。

以下是这些要点的摘要：

- 尽可能避免屏幕空间效果；最好的情况是它会看起来不好，最坏的情况是会导致眼睛的立体错位。
- 目标帧率略高于目标显示帧率；这样可以保证任何的卡顿都不是导致不适的主要原因（不要依赖重投影技术）。
- 如果需要增加帧率，可以考虑降低渲染分辨率。在 UE4 中，可以使用控制台命令 r.screenPercentage（参见第 10 章）。
- 玩家的头部运动应该始终在摄像机的控制之内。避免使用电影摄像机，在菜单或者关卡加载过程中使用异步加载。避免摄像机抖动和其他非玩家控制的摄像机移动效果。
- 不要遮挡玩家视野区域；在旋转时会引起不适。
- 不要加速；这样会导致眼睛前庭错位（玩家看到的和听到的不匹配）及恶心（想要了解更多信息，请参阅第 9 章）

与之前一样，这两个文档包含了对任何 VR 开发者都很重要的信息。建议把它们都读一遍。

1.3 小结

现在你已经熟悉了本书中用到的一些术语，包括 UE4 支持的大量 VR 硬件设备，一些用于创建 VR 体验的库（即使不能直接与它们连接）。此外，你也已经了解了用于优化虚拟现实体验的一些技术（如果想要学习更多内容，请参阅第 10 章），以及在 VR 中两个主流公司的最佳实践。